V.E.S.
Since 1962

Vivekanand Education Society's
Institute of Technology

(An Autonomous Institute Affiliated to University of Mumbai)

Department of Information Technology

A.Y. 24-25

Advance DevOps Lab

Experiment No.	Assignment-2
Title.	Assignment - 02
Roll No.	23
Name	Mohit . S . Kerkar
Class	D15 C
Subject	Advance DevOps
Lab Outcome	LO6:To engineer a composition of nano services using AWS Lambda and Step Functions with the Serverless Framework
Signature:	
Grade:	05

Name : Mohit S. Kefkar Div: D15C Roll NO: 23.

Adv. DevOps Assg. 2:

Q1. Create a Rest API with serverless framework.

Ans Creating REST API with serverless framework is an efficient way to deploy serverless applications that can scale automically without managing server.

(i) serverless framework: A powerful tool that deployment of services and serverless applications across various cloud providers such as AWS, Azure and Google Cloud.

(ii) Serverless Architechture: This design model allows developers to build applications without worrying about underlying infrastructure, enabling focus on code and business logic.

(iii) REST API: Representional State Transfer is architecture style for designing network applications.

Steps for creating REST API for serverless framework.

(1) Install a serverless framework.

We start by installing serverless framework CLI globally using node package manager (npm). This allows us to manage serverless applications directly from our terminal.

(2) Creating a Node.js serverless project.

A directory is created for our project, where we will initialize a serverless service project. This service will have all our lambda functions, configurations and child resources. Using the command serverless create, we ~~your~~ setup a template for AWS Node.js microservices to be ~~that~~ deployed to AWS Lambda

(3) Project Structure

The project scaffold creates essential files like handler.js (which contains code for Lambda functions), and serverless.yml,

④ Create a REST API resource.
In the serverless.yml file we define function that hand
post requests of HTTP.

⑤ Deploy the service
With the 's/s deploy' command serverless framework
packages your applications, uploads necessary resource
to AWS and sets up the infrastructure.

⑥ Testing the API: Once deployed, we can test REST
using tools like curl or Postman by making post
requests to generated API.

⑦ Storing data in DynamoDB: To store submitted
candidate data you integrate AWS DynamoDB as a dat

⑧ Adding more functionalities: Adding functionalities lik
'list all candidates, get candidates by ID'.

⑨ AWS IAM Permissions: We need to ensure that
serverless framework is given right permission to inter
with AWS resources like DynamoDB.

⑩ Monitoring and Maintainence: After deployment
serverless framework provides service information
like deployed endpoints, API key, log streams.

Q2. Case study for SonarQube.

~~Ans~~ Creating our own profile on SonarQube for testing
project quality. Use SonarQube to analyze your
Github code. Install Sonar lint in your Java Intell
IDE and analyze java code. Analyze python project
with SonarQube. Etzica

Ans SonarQube is an open source platform used for
continuous inspection of code quality. It detects bugs, code
smells and security vulnerabilities in project across vario

programming languages.

1. Profile section in SonarQube..

Quality profiles in SonarQube are essential configurations that define rules applied during code analysis. Each project has a quality profile for every supported language with default being 'sonarway' profile comes built in for ~~default~~ all languages. Custom profiles can be created ~~f~~ by copying or extending existing ones. Copying creates an independent profile, while extending inherit rules from parent profile and reftects future changes automatically. We can ~~set~~ activate or deactivate rules, prioritize contain rules and configure parameters to tailor profile to specific projects. Permissions to manage a quality profile are restricted to users with administrative priviledges. SonarQube allows for the comparison of two profiles to check for differences in activated rules and users can track changes via event ~~tag~~ log. Quality profiles can also be imported from other instances via backup and restore. To ensure profiles include new rules its important to check against updated built in profiles or ~~s~~ SonarQube rules page.

2. Using SonarCloud to analyze GitHub Code:

SonarCloud is cloud-based counterpart of SonarQube that intergrates directly with GitHub, BitBucket, Azure and GitHub repositories. To get started with SonarQube via GitHub organization or personal product page and connect your GitHub organization or account. Once connected, SonarCloud mirrors our GitHub setup with each project corresponding to GitHub repositories. After setting up ~~each~~ organization choose subscription plan (for public repository) . Next, import repositories into SonarCloud organization where each GitHub repositor becomes a SonarCloud project. Define 'new code' to focus on recent changes and ~~at~~ choose between autom analysis or CI based analysis. Automatic analysis happ directly in ~~&~~ SonarCloud, while CI based analysis integrate with your build process once the analysis is complete results can be viewed in both Sonar-Cloud and GitHub including security import size

3. SonarLint in Java IDE.

SonarLint is an IDE that performs on-the-fly code analysis as your write code. It helps develops detect bugs, security vulnerabilities and code smells directly in the development environment such as IntelliJ Idea or Elipse. To set it up, install the SonarLint plugin, configure the connection with SonarQube or SonarCloud and select the project profile to analyze Java code. This approach ensures immediate feedback on code quality, promoting clean and maintainable code from beginning.

4. Analyzing Python Projects with SonarQube.

SonarQube supports Python test coverage reporting but requires third party tool like coverage.py to generate the coverage port. To enable coverage adjust our blind process so that coverage tool runs before Sonar scanner and ensures ~~tool~~ report file is saved in a diffrent pa

For setup, ~~you~~ we can Tox, Pytest and coverage.py to configure and run test. In our tox.ini include configurations for py-test and coverage to generate coverage report in XML format. The build process can also be automated using SonarQube scan. Ensure report in ~~to~~ Coberurata XML format and place where scanner can access it.

5. Analysing Node.js projects with SonarQube.
For Node.js projects SonarQube can analyze Javascript and Typescript code. Similar to the python setup, you can configure SonarQube to analyze Node.js projects by installing the appropriate plugin and using SonarScanner to scan the project. SonarQube will check the code against Industry standard rules and best practices, flogging issues related to security vulnerabilities bugs and performance optimization.

3 At a large organisation, your centralized operations team may get many repetitive infrastructure requests, you can use Terraform to build a "self-serve infrastructure requests, you can use Terraform to build a "self serve" infrastructure model that lets product teams manage their own infrastructure independently. You can create and use Terraform modules that codify the standards for deploying and managing services in your organizations, allowing teams in efficiently deploy services in compliance with your organization pratices. Terrafom Cloud can also integrate with ticketing system like service. Now to automatically generate new infrastructure requests

Ans Implementing a 'self aware' infrastructure model using Terraform can transform how large organisa manage their infrastructure independently, organisa can enhance efficiently, reduce bottlenecks, and ensure compliance with established needs.

- The need for self service infrastructure.

In large organisation, centralized operations term often fall on overwhelming number of repetitiv requests. This can lead to delays in service delive and frustration among product teams who need to move quickly. A self-service model allows teams to provision and manage their infrastructure without relying on the operations term for every request.

- Benefits of using Terraform

1. Modularity

→ Terraform modules encapsulates standard configurations for various infrastructure components (eg. network, databases, compute resources)

→ Teams can reuse these modules across different projects, reducing redundancy and minimizing risk of error.

2. Standardization.

→ By defining best practices within modules, organizations can ensure that all deployments comply with internal policies and standards.

→ This consistency helps maintain security and operational integrity acro the organisation.

3. Increased efficiency.
→ Product teams can deploy services quickly by using pre-defined modules, significantly reducing the spent on infrastructure setup.
→ This allows term to focus on developing rather than managing, infrastructure.

4. Integration with ticketing systems.
→ Terraform Cloud can integrate with ticketing systems like service. Now to automate the generation of infrastructure requests.
→ This integration streamlines workflows by allowing teams to initiate requests directly from their ticketing platform, reducing manual intervention.

Here are the implementation steps.

1. Identify Infrastructure Components.
- Begin by identifying which components of your infrastructure can be modularized (eg. VPCs, security groups, load balancing)

2. Develop Terraform modules.
- Creates reusable modules that define the desired configuration and resources.
- Ensure each module include input variables for customization and outputs for integration with other modules.

3. Establish Governance and Best Prastices.

- Define guidelines for modules usage, versioning an documentation to ensure clarity and maintainability
- Encourage terms to contribute to module developme and share improvements.

4. Testing and validation:

- Implement a testing framework to validate to module fun before development.
- Use tools like terraform plan to preview changes and catch potential issues early.

Best pratices for module management.

- Utilize the terraform registry.
Leverage existing community modules from the Terraform registry to avoid reinventing solutions and ensure adherence to best practices
- Version control: Implement versioning for your modules to track changes overtime. This helps manage dependencies effectively and minimize description during updates.
- Documentation: Maintain comprehensive documentation to each module including usage examples, input/o descriptions and any dependencies.
- Encourage collaboration: Foster a culture of collaboration by sharing modules across teams. This promotes consistency in deployments and facililates knowledge within the organizations.

By adopting a self service infrastructure model with Terraform organization can empower product teams to efficiently managge their own infrastructure while ensuring compliance with established standard.